Docket No. 4020

SPECIFICATION

TITLE: EMERGENCY VEHICLE TRAFFIC SIGNAL PREEMPTION SYSTEM

This application is a Continuation-In-Part of Application Serial No. 10/642,435, filed August 15, 2003, and Application Serial No. 60/403,916 filed August 15, 2002. The invention described herein was made in the performance of work under a NASA contract and is subject to the provisions of Public Law 96-517 (U.S.C. 202) in which the Contractor has elected to retain title.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to systems for controlling vehicle traffic signals to allow safe passage of emergency vehicles and more particularly relates to a system for autonomously preempting traffic signals at an intersection that includes a vehicle transponder, a real-time intersection controller and monitor (with an intersection-based visual and/or audio alarm warning system), an operations display and control software, and a wide-area communications network.

2. Background Information

Present systems used to preempt traffic signals and clear intersections for emergency vehicles responding to a life-saving event often come with severe limitations. They rely on:

Express Mail Label Number: EV449028019US
I hereby certify that this correspondence is being deposited with the U. S. Postal Service as EXPRESS MAIL, postage prepaid, under 37 CFR 1.10 in an envelope addressed to the Commissioner of Patents, P. O. Box 1450, Alexandria, VA 22313-1450.
Date: March 24, 2004
By:
David O'Reilly, Reg. No. 26,102

sound activation, optical activation, direct microwave activation, and a combination of all the above. All of these systems have severe operational limitations affected by weather, line of sight, and critical range. These systems often have further drawbacks requiring them to be activated by the emergency vehicle operator or first responder (herein referred to as "e-operator"). These systems also severely disrupt the normal phasing patterns of a traffic controller's nominal programming because these systems do not provide real-time monitoring of intersection phases or timing.

Emergency vehicles currently rely on vehicle horn, sirens, and flashing lights to prevent accidental collisions with pedestrians or other vehicles at intersections. E-operators must focus all their attention on driving the vehicles. Other preemption systems fail to provide visual or audio feedback systems (to either motorists or e-operators) that are physically located in the intersection (herein referred to as "intersection-based warnings"). Such preemption systems compromise motorist and e-operator safety, as there is no awareness of a traffic-light preemption event (referred herein as "silent preemption"). Additionally, these systems fail to provide real-time feedback to e-operators through warning devices inside their vehicles (herein referred to as "vehicle-based warnings"). These factors have the effect that e-

operators do not get the feedback required and soon stop using the system.

An intersection-based preemption system that provides feedback and is activated autonomously by an approaching emergency vehicle is needed. Such a system overcomes some of the drawbacks of available systems. Intersection-based visual warnings are proven effective for motorists, and are also critically important to e-operators when multiple emergency vehicles are approaching the same intersections (referred herein as "conflict detection"). These displays are directly in their field-of-vision and e-operators are immediately aware of potential conflicts. Human factors studies often refer to such indicators as "real-world". Intersection-based warnings combined with autonomous activation removes the distraction by keeping drivers' eyes on the road.

A system is needed that takes special consideration of pedestrians. Visual intersection-based warnings may fail to get the attention of pedestrians standing near an intersection. For this reason, audible alerts in addition to visual may be the most effective (and rapid) warning system of the approach of emergency vehicles. There is also the difficulty that pedestrians may often be in harms way if they fail to hear an approaching emergency vehicle. Although vehicle sirens are especially loud, many circumstances can lead to dangerous

situations and potential injury. For instance, an especially long crosswalk may take up to 20 seconds to cross. In that time, an emergency vehicle may be heard, perhaps stranding the pedestrian in the middle of a crosswalk. Likewise, in extremely busy metropolitan intersections, ambient noise in the building occlusions may prevent warning of the emergency vehicle until just seconds before the vehicle arrived at an intersection. A system is needed that disables normal pedestrian clearance at intersections long before actual preemption has been triggered (herein referred to as "pedestrian-inhibit"). This system would greatly enhance the safety of emergency vehicle preemption by preventing pedestrians from entering an intersection long before a vehicle arrives (or can be seen or heard).

Existing preemption systems provide little or no visibility, configuration control, or remote interaction with their operation or function. A system is needed that provides real-time feedback, monitoring, logging, and control of vehicle and intersection preemption-related data. This data would be displayed at both mobile stations and central operation center(s). Additionally, a system is needed that provides secure, robust transfer of data to/from intersections, vehicles, and operation center(s) using either wireless or LAN architectures. All of these functions enable logistical

commanders and traffic management authorities to coordinate, configure, and monitor activity in the overall preemption network.

It is one object of the present invention to provide an emergency vehicle traffic signal preemption system that is fully autonomous and not dependent on the intersection being in visual range.

Still another object of the present invention is to provide an emergency vehicle traffic signal preemption system that includes a real-time monitor of intersection phase to optimize triggers and timing for both preempt and pedestrian-inhibit functions. This includes minimizing disruption of normal traffic controller behavior and sequencing.

Still another object of the present invention is to provide an emergency vehicle traffic preemption system that includes visual displays in the intersections (and interfaces to such displays) indicating direction and location of approaching emergency vehicle(s).

Still another object of the present invention is to provide an emergency vehicle traffic signal preemption system that provides conflict detection (between emergency vehicles and e-operators) and alerts other emergency vehicles in the area. This conflict detection is provided in two forms: intersection-based warnings and vehicle-based warnings.

Still another object of the present invention is to provide an emergency vehicle traffic signal preemption system that includes a pedestrian audio warning signal to supplement the intersection-based visual display and the audio signals from emergency vehicles.

Yet another object of the present invention is to provide an emergency vehicle preemption system having an autonomous emergency vehicle transponder including an on-board diagnostic (OBD) interface, a real-time navigation interface and position estimation module, and a communications monitor and control interface.

Still another object of the present invention is to provide an emergency vehicle traffic signal preemption system that allows real-time remote access, monitoring, and tracking of the entire preemption system via secure wide-area networks (wireless and LAN). This includes access to the operations display and control software (herein referred to as "operations software") from management centers (TMC, 911-call center, etc.), mobile commanders, as well as individual emergency responder vehicles.

BRIEF DESCRIPTION OF THE INVENTION

The purpose of the present invention is to provide an improved emergency vehicle traffic signal preemption system including autonomous operation, real-time phase monitoring and

visual/audio signals to alert motorists and pedestrians of the approach of emergency vehicles.

The system is fully autonomous and is not affected by range, weather, or line of sight. It provides real-time monitoring of the intersection phases to optimize intersection timing and provide the visual display to alert motorist of oncoming emergency vehicle and the direction it is coming from. This system is an improvement for use with the system disclosed and described in U.S. Patent No. 4,704,610 of Smith et al issued November 3, 1987 and incorporated herein by reference. The system also provides an added feature of conflict indication inside the emergency vehicle operator, indicating that another emergency vehicle is responding and is approaching the same intersection, indicating which vehicle has the preemption and right of way.

This system is unique in that it is fully autonomous and not dependent on the intersection being in visual range. It provides conflict detection and alerts other emergency vehicle operators in the area, has the ability to interrupt pedestrian access, stops preemption when an emergency vehicle stops, and provides interface to and control of the system disclosed and described in the above-identified patent.

The improved emergency vehicle traffic signal preemption system consists of three major subsystems. An intersection

monitor and control, an emergency vehicle transponder and its interfaces, and a wide area communications network and its associated proprietary control program software. The emergency vehicle intersection preemption design connects intersections and vehicles over a two-way wide area wireless communications network. This network is synchronized via Global Positioning System (GPS) timing signals. The system is also capable of using existing traffic management LAN networks to relay data to operations center(s).

When an e-operator receives an emergency response request, the vehicle is placed in a priority-code (i.e. Code-3) mode with lights and sirens operating. The vehicle emergency state is read via an emergency-code vehicle interface. At the same moment, the vehicle preemption transponder reads the vehicle on-board diagnostics (OBD) data and determines speed and acceleration, and gathers navigation data from one of several navigation systems. This data is collected by an on-board microprocessor that processes this information and predicts heading and position. Estimation techniques include (but are not limited to) dead reckoning and position hysteresis - historical dependence - and are dependent on the sensor data quality. This information is then formatted, the vehicle identification (ID) and absolute time added, and the data is then transmitted to various both intersections and vehicles

within the design area of coverage. The data is also immediately forwarded along the network to subscribing mobile and fixed operations center(s).

Intersection processors receive the data, identify the vehicle's estimate-time-of-arrival (ETA), and compare it with other vehicles possibly approaching their locations. It then determines which vehicle obtains highest priority (depending on location history, priority-type of vehicle, and other factors). The processor sends notification to all approaching emergency vehicles, warns of any potential conflict, and notifies the local e-operators which vehicle has the right of way.

Simultaneously the processor collects real-time intersection phasing and timing information and calculates when preemption should start based on the vehicle(s) ETA. The system includes the real-time monitoring of analog, digital, and stand-alone (disabled monitoring) controllers. This monitoring optimizes preempt behavior and provides a closed-loop verification that preempt commands are executed by the intersection controller.

It also calculates when to trigger the pedestrian-inhibit function to prevent clearance for crossing access. When preemption starts, intersection-based warning displays are sent coded commands via a wireless or hard-line connection to light the proper icons. For each direction, the displays show all

preempting emergency vehicles' direction and location, and light the appropriate emergency vehicle message (i.e. "Warning Emergency Vehicle"). All this takes place in real time, in a manner appropriate to insure an intersection is preempted early enough for safe and clear access, and in such a way as to minimized speed reduction for the emergency vehicles.

The system disclosed herein provides a number of improvements of the above-identified patent. It is an autonomous system that does not need involvement of emergency vehicle operator. It also includes expanded system capabilities using emergency vehicle on-board diagnostics (OBD), monitoring multiple emergency vehicles approaching the same intersection using Global Positioning System (GPS), and speed and heading information for multiple emergency vehicles to determine the right of way. An intersection status is transmitted to emergency vehicle dashboards indicating when the intersection is safe to traverse. A dashboard display indicates to the vehicle operator the status of an intersection. The system is also capable of providing dynamic and customized displays via an interface to the vehicle-based PC (personal computer) systems. This interface provides detailed, real-time positioning and status of all neighboring emergency vehicles and intersections. It allows e-operators to view maps with active vehicles and also allows for enhanced

conflict detection notification. The system also includes a wide area wireless RF communication links between emergency vehicles and intersections. This system is reliable and unaffected by weather, rain, or lack of line of sight.

Simultaneous to preemption triggers, pedestrian audio alerts are activated when emergency vehicles are approaching an intersection. These are important because often visual signs at an intersection may not be clearly visible to a pedestrian. Beepers, bells, sirens, or even spoken instructions at high volume can be used.

Several types of emergency vehicle location and navigation information retrieval are possible. Among these are Global Positioning Systems (GPS), dead reckoning, beacon triangulation, tags, traffic loop, RDIF, etc. Each vehicle has an identification (ID) tag that allows transmission to the appropriate vehicle that it has the right-of-way to a preempted intersection.

The improvements to the existing system in the above-identified patent are to enhance the performance but the purpose of the system remains the same. That is, to alert and stop vehicles and pedestrians from using an intersection and to allow an emergency vehicle to pass safely. Some prior warning is necessary to allow clearing the intersection. The previous implementation uses a one-way infrared link to transmit

approach and departure information of emergency vehicle to the intersection which is equipped with four emergency vehicle status display panels mounted next to the usual traffic lights at each intersection.

The system transmits a signal causing all traffic lights at an intersection to switch to "red" thus stopping all traffic in all directions. In addition, the display panels flash a relatively large "emergency vehicle" therein with a graphic display indicating the lane and direction of traffic taken by an emergency vehicle. The range of the infrared transmitter can be as much as 1,000 feet allowing sufficient time to clear the intersection. The new improved system utilizes a wide area wireless RF two-way communication link between emergency vehicles and intersections. This method is more reliable and not affected by weather, lack of line of sight, range limitation or obstructions.

Another advantage of the two-way wireless RF communications link between the intersections and emergency vehicles is the ability to display much more useful data in the vehicles helping the vehicle operator maneuver his vehicle most efficiently and safely. This data includes (but is not limited to) emergency-code levels, vehicle acceleration, vehicle type, and vehicle health. This method also enables feedback communication to be sent from the intersections to the

vehicles, providing vehicle-based warnings (or confirmation) of system activity. Intersection "green" status shows when an intersection has been preempted and priority is given to the receiving vehicle, allowing safe passage. If more than one emergency vehicle approaches an intersection, the system determines which vehicle should have the right of way depending on location information (GPS, traffic loop, beacon, etc.), direction and speed sent to the intersection control. A proprietary control program determines the right of way and sends the result to emergency vehicles. The encrypted data package transmitted over transceivers is tagged with the vehicle ID and time to insure proper and certified utilization.

Another improvement to the system is an audio warning system intended to alert pedestrians that an intersection has been preempted and must be kept clear. One desirable implementation would utilize loudspeakers mounted near the four corners of the intersection where pedestrians normally gather to cross. A spoken message, such as "warning, emergency vehicle approaching, do not walk", may be most preferred but any audible signal such as a wailing sound, a siren, or any other familiar emergency sound may be utilized.

Another goal of the improved system is creation of an autonomous system that is activated by reception of a priority-code (i.e. Code-3) status or alarm. The operator of the

emergency vehicle can concentrate on his primary duty which is to arrive at the sight of the emergency safely in the shortest time possible without worrying about the activation of the system. A priority-code starts the process of communication between an intersection that is being approached and the emergency vehicle and the system performs the functions described above. Also, both vehicle-based warnings and intersection-based warnings provide positive feedback that an e-operator has secured an intersection. This directly translates into a reduction of emergency workers' stress levels.

The information available from the emergency vehicle and intersection controllers may be transmitted to a central location such as a dispatch center or traffic control center to display the status of multiplicity of intersections and emergency vehicles. Such information being displayed on a status board can be invaluable in managing emergency situations (especially large-scale incidents) in a more sufficient manner because it makes available information on a real-time basis for the officials in charge. Commands and configuration information can also be sent back to intersections and vehicles to instantly meet changing needs or requirements. These instructions can include the creation of large emergency corridors (herein referred to as an "e-corridor") whereby a

series of sequential intersections are preempted in the same direction.

The above and other objects, advantages, and novel features of the invention will be more fully understood from the following detailed description and the accompanying drawings, in which:

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of the functions of intersection hardware for the emergency vehicle traffic signal preemption system (herein referred to as "preemption system"), as used for interfacing with all intersection controllers.

Figure 2 is a block diagram of the functions in an emergency vehicle transponder for the preemption system.

Figure 3 is an example schematic block diagram of a standard vehicle transponder for the preemption system.

Figure 4 is an example schematic diagram of a vehicle on-board diagnostic (OBD) circuit for the preemption system.

Figure 5 is a functional organizational diagram of the three major subsystems for the preemption system.

Figure 6 is a schematic block diagram of the intersection hardware for the preemption system, as configured for interfacing to an intersection controller without monitoring.

Figure 7 is a schematic block diagram of the intersection hardware for the preemption system, as configured for

interfacing to an intersection controller with digital BUS monitoring.

Figure 8 is a schematic block diagram of the intersection hardware for the preemption system, as configured for interfacing to an intersection controller with analog monitoring.

Figure 9 is a general flow diagram of the intersection control program software for the preemption system.

Figure 10 is a general flow diagram of the vehicle transponder control program software for the preemption system.

Figure 11 is a detailed decision flow diagram of the preempt monitor task component for the intersection control program software.

Figure 12 is a detailed time sequence diagram of the standard preemption criteria used by the intersection control program software in a typical preemption scenario.

Figure 13 is a layout and topology diagram of the communications and operations network for the preemption system.

Figure 14 is a block diagram of the functions and data flow of the operations software for the preemption system.

Figure 15 is an example of the data status module display component and alerts module display component, used in the operations software for the preemption system.

Figure 16 is an example of the intersections module display component, used in the operations software for the preemption system.

Figure 17 is an example of the vehicles module display component and the mapping module display component, used in the operations software for the preemption system.

DETAILED DESCRIPTION OF THE INVENTION

The three major subsystems in the emergency vehicle traffic signal preemption system are shown in Figure 5: the vehicle transponder 200, the intersection hardware 230, and the communications and operations network 260.

The vehicle transponder 200 is composed of three main components. First, the vehicle computer interface module 205 includes the on-board diagnostics circuit and the emergency priority code interface. Second, the navigation predict module 210 uses navigation sensors such as GPS and INU (inertial NAV unit) sensors to generate both absolute and estimated dead reckoning position reports. Third, the transponder control module 215 provides an interface to the e-operator via LEDs, PC display, or PDA device.

The intersection hardware 230 is composed of three main components. First, the intersection monitor module 235 provides real-time reading and logging of controller signal and pedestrian phasing and timing. Second, the intersection

control module 240 performs ETA calculations using vehicle positions and local known mapping topology (commonly known as map-matching). This module also tracks and logs vehicles, actuates and verifies preempt signals, manages communications between other networked units, and manages remotely-generated intersection configuration commands. Third, the warning alerts control module 245 actuates intersection-based visual and/or audio warnings. This module also ensures that warning alerts follow specific rules and timing parameters that govern the sequencing of warning signs with traffic lights.

The communications and operations network 260 is composed of three main components. First, the slave (end-unit) transceivers in vehicles and intersections 275 relay the core preemption status and configuration data to the backbone network. Second, the backbone wireless or LAN network 270 is a hybrid wide-area network designed to route data between mobile wireless vehicles, hard-lined and isolated wireless intersections, and the central operation center(s). Third, the operations software 265 provides for display of all real-time data generated by the intersections and vehicles including positions/speed, phasing, preemption-status, vehicle diagnostics, logged information, configuration data, and many other data parameters. This display/control software 265 can be mobilized for use in any management center, staging area, or

even an entire fleet of emergency vehicles.

The functional details of the major subsystems in the emergency vehicle traffic signal preemption system are illustrated in the block diagrams of Figure 1, Figure 2, and Figure 13. Figure 1 illustrates the functional details of the system at each intersection, Figure 2 illustrates the functions of the system installed in an emergency vehicle, and Figure 13 illustrates the topology and display/control software used for the communications and operations network.

Traffic light control system 100 at an intersection includes traffic light controller 20 (housed in cabinet 500) that generates the appropriate sequence of on-time and off-time for the various traffic lights that controls vehicular and pedestrian traffic at an intersection. Traffic light controller 20 also has the capability to be forced by external signals into a mode that activates "green" lights in a specified direction and "red" lights in all other directions, allowing safe passage for emergency vehicles from the "green" direction. Controller 20 is preferably a micro-processing circuit driving isolated lamp drivers but discrete designs are also feasible. Some intersections may be more complicated, controlling turn lanes with arrow lights, but the basic principles remain the same.

An example of an intersection being controlled by the

system and functions disclosed and describe herein is shown in Figure 1 of U.S. Patent No. 4,704,610 referred to hereinabove and incorporated herein by reference. This figure shows the signage and approach of emergency vehicles being controlled. The only feature missing is the pedestrian control signs at each corner which are an added feature of the invention disclosed and described herein.

Traffic light controller 20 generates signals to control pedestrian lights 22a, 22b, 22c, and 22d and also controls the operation of traffic lights 24a, 24b, 24c, and 24d. An intersection having traffic lights can be connected to a system using the emergency vehicle preemption system by addition of the functions described hereinafter without the need to rebuild an existing installation.

The heart of the additional equipment is the intersection control module, a microprocessor 515 (e.g., a ZWorld LP 3100 CPU) operated by proprietary control program software 35. Controller 10 (housed in hardware module 510) receives information from emergency vehicles that approach an intersection via wireless RF transceiver 40 and antenna 41. This information contains data about the predicted position, heading, other navigation data of the emergency vehicle, and its priority-code status 36 (i.e. Code-3, Code-2, or other) thus notifying the intersection of its relative location.

Figure 9 illustrates the general functionality of the intersection control program software and firmware 35 (see Appendix B). The vehicle monitor software task 605 running on the intersection CPU 515 tracks all local vehicles and maintains a log of all activity. The task also sends conflict detection warnings, when appropriate, to the vehicles.

The intersection control program 35 continually evaluates its preemption rules as vehicle updates are received. Position and priority parameters of each vehicle within range are analyzed by the intersection preempt monitor software task 600. The primary decision logic of this task is illustrated in Figure 11. Appendix A provides detailed explanations of the terms and parameters used in this figure and the description below. The preempt monitor task uses map-matching techniques to evaluate all vehicles against all eligible cross street segments 700 to determine which vehicles are inbound or outbound 730 from the intersection. The task assigns preemption priority to that vehicle which is within critical perimeter zones (pedestrian 705 and preempt 706), in high priority priority-code 710, and is a valid vehicle type 720. In order to optimize the preemption process, it compares the minimum vehicle-ETA with both the intersection clearance time (time-to-preempt) and a minimum complete-preemption time (threshold) 715.

Figure 12 provides a visual illustration of the logic of the intersection preempt monitor software task. The diagram shows the actual positions ($p_\#$) based in time along the actual path 621 of the vehicle. For every actual position ($p_\#$), there is a same-time position report ($e_\#$) along the estimated path 620 of the vehicle. For instance, p_1 623 and e_1 622 both occur at same time t_1. The diagram illustrates the estimate path 620 with valid position-lock (i.e. GPS occlusion), as well as temporary loss of position-lock 624 when dead reckoning is used to compensate. The diagram also illustrates the multiple uses of proximity (perimeter) layers, with a pedestrian-inhibit perimeter 625 ("max-PED-perimeter"), a preemption-allowed perimeter 626 ("max-preempt-perimeter"), a critical distance perimeter 627, and multiple critical distance street segments 628. Non-critical segments 636 are also shown (these street segments require additional evaluation based on vehicle-ETA). The exit window 631 displays an example exit distance range where egress intersection-based warnings are allowed to be activated (based on configurable minimum and maximum exit distance criteria). Also, the evaluation of vehicle heading compared against the road heading is shown as the direction-error 622. The acceptable deviation of the estimated position from the center-line of the street 630 is also shown.

Figure 12 also shows one of the more advanced preemption

techniques used on the intersection control program, the use of "threshold-lag" 640, 641, and 642. "Threshold-lag" is defined in Appendix-A. In simple terms it is percentage error factor added to the threshold that gives the "benefit-of-the-doubt" to any actively preempting vehicle. Initially (prior to preemption), the threshold-lag factor 640 is zero percent (0%). When the threshold is crossed, the threshold-lag becomes its maximum value (i.e. 30%), and it is added to both the threshold-time and the time-to-preempt factors for comparison to vehicle-ETA. Once a vehicle has crossed the threshold, and the threshold-lag has been expanded, the threshold-lag linearly decreases back to zero percent (0%) over a small period (i.e. 10 seconds). This calculation is just one form of hysteresis (historical dependence) techniques used in the invention.

Figures 6, 7 and 8 are schematics that show detailed layouts of the intersection hardware components and, most specifically, multiple configurations for real-time monitoring of phasing/timing controller signals. The configuration in Figure 7 provides for interfacing to digital BUS intersection controllers 20b (such as NEMA TS1 controller models). The configuration in Figure 8 provides for interfacing to analog-based intersection controllers 20c (such as type 170 controller models). On such analog systems, traffic lights signals are monitored by a fail-safe, isolated, high impedance tap and

subsequent digital circuit processing. The monitor data is available for remote monitoring via the wide area communications and operations network. As shown in Figure 6, the system is still compatible with controllers that disable monitoring 20a or where monitoring is not desired.

Real-time monitor information is read and analyzed by the intersection monitor software task 610. These calculated values are forwarded to the preempt monitor 600, where these intersection phasing values are integrated with real-time vehicle information. The software attempts to optimize preempt triggers with "time-to-preempt" calculations and "time-to-pedestrian-inhibit" calculations, as compared to the ETA of all approaching emergency vehicles. The goal is to provide minimal disruption to the nominal controller behavior and to maximize the throughput of emergency vehicles through the preemption intersection network. Also, unlike other preemption systems, beyond simply sending a preempt command (actuating a preempt signal), the real-time monitor independently measures the state of the controller-actuated traffic light signals. This provides a critical closed-loop design: it assures that preempt commands are actually executed.

Real-time status monitor 42 is unique because it verifies the state of the traffic signals and sends the intersection status (i.e. "intersection preempted", "conflict detected", or

"no preemption") to intersection control module 10. That is, real-time status monitor receives (i.e., "reads") the output from traffic light controller 20 and pedestrian lights 22a through 22d and traffic lights 24a through 24d and transmits that information to intersection control module 10. Intersection control module 10 in turn relays that information to emergency vehicles via wireless RF transceiver 40 and antenna 41. Intersection control module 10 now sends signals to emergency display panels 45a, 45b, 45c, and 45d to light and flash large emergency signs with the proper icons at each corner of an intersection showing the position of any approaching emergency vehicle relative to the traffic lanes of the intersection as shown and described in the above-identified U.S. patent incorporated herein. The display panels 45a-45d and proper icons used at each corner of an intersection are shown in Figure 2 of the U.S. patent referenced hereinabove. The signage is also illustrated in U.S. Design Patent No. 305,673, issued January 23, 1990, and also incorporated herein by reference.

Also, the real-time status monitor 42 provides which is transmitted via RF master transceiver (or LAN) 60 and antenna 61 to a central monitoring system such as a dispatcher's office. Reciprocally, the intersection receives information on the state of its neighboring intersections. This closed-loop

architecture allows various units in the network to accurately predict future movement, log critical information, and notify users of the system state.

The intersection control program 35 (specifically the preempt monitor software task 600) uses map-matching techniques to compare vehicle navigation and position estimates with the approach paths (cross-streets stored locally as map vectors). This way the intersection can determine if any vehicle is on an inbound course towards the intersection by "snapping" it to the closest street. As an example, one of the calculations is the "critical distance" test. This evaluates whether an approaching car has statistically committed itself to crossing through the local intersection based on lack of turning options. Because of the knowledge of the road map, the intersection can preempt even when the "critical distance" is not line-of-sight. As an additional example, in the event that any vehicle comes with a "warning distance" of the intersection (1000-ft commonly used), the control program 35 will actuate pedestrian-inhibit functions. Pedestrian lights 22a through 22d are changed to prevent pedestrian traffic. Through a combination of hysteresis-based (historical dependence) algorithms and dynamic proximity "windows", the system is able to optimally route emergency vehicles across the map grid. It is also able to effectively mitigate lossy communications,

lossy navigation data, and other unpredictable delays in the system.

Another improvement to the system is the provision of an audio warning to pedestrians. Thus simultaneously with controlling the lights and pedestrian flashing signals, controller 10 generates an audio message to be delivered from audio warning device 50 to speakers 51a through 51d.

As mentioned, the details of the software in the intersection control program for implementing the functions of the system are provided in Appendix B. Because the functions controlled are described in great detail in the text, many software solutions to implement the functions will be apparent to those skilled in the art.

Emergency vehicle functions for the preemption system are illustrated in the block diagram of Figure 2. A transponder box 99 (and cables 98, 98a) are installed in each emergency vehicle and provide the functions that facilitate communication with preempt-able intersections, other emergency vehicles, and also central monitoring stations such as a dispatching center. Inputs and outputs to and from the emergency vehicle system are handled by transponder control module 30 under the direction of proprietary control program software 15. Vehicle parameters are determined from several inputs provided to transponder control module 30.

Vehicle position is available from GPS receiver 38 via antenna 39. Several positioning inputs 96 are available from ports in navigation input device 34. Optional alternative inputs from ports and navigation input device 34 are INU (inertial navigation and estimation unit 29) parameters including accelerometers, gyroscopes, wheel-tachometers, and heading indicators. Other inputs include ID tag tracking, beacon triangulation, modified traffic loop detectors, and others. Vehicle information such as speed and acceleration are read in real-time from the vehicle computer 33 using the on-board diagnostic (OBD) interface cable and connector 33a. These signals are converted and verified by the OBD circuit board 32 and the translated digital signals are input to transponder control module 30 (embedded on a micro-controller 97).

The emergency vehicle transponder system communicates with intersections via wireless RF transceiver 44 and antenna 45. The vehicles and intersections software task 670 running on the vehicle transponder handles incoming intersection preempt alerts and vehicle position reports from nearby units. It receives feedback verification and displays the information on-board by activating one or more LEDs 56, 57, or 58 on the LED display 54. If it receives a signal for safe passage through an intersection, "green" LED 56 is illuminated. If another

high-priority emergency vehicle is concurrently trying to preempt the same intersection, "yellow" LED 57 is illuminated. Illumination of "red" LED 58 indicates that there is no preemption at the intersection. LEDs 56 through 58 are driven by "intersection preempted" logic circuit 55. Logic circuit 55 can also provide customized outputs to dynamic display devices 59, such as PC monitor displays (LCD's) and Personal Digital Assistants (PDA's). Such devices are commonly used for law enforcement applications within the vehicle. As mentioned, the operations software shown in Figure 14 can be mobilized 80 and run on any vehicle-based auxiliary hardware device with a standard operating system. The vehicle interface software task 665 in the transponder control program allows advanced mapping and alerting of active nearby intersections and vehicles.

Emergency vehicle status is available in real time via master RF transceiver 64 and antenna 65 to a central monitoring station. Thus the position of any vehicle as well as the status at an intersection is always available at some centrally located dispatch station.

As indicated previously, the software in control program 15 to implement the functions of the transponder described above has many possible solutions. Thus the software provided to control the operation of transponder control module 30 can be designed and implemented by anyone skilled in the art given

the detailed explanation of the system and functions described hereinabove. Also, as previously mentioned, Appendix B provides detailed pseudo-code of a full-featured version of the software for both the intersection and vehicle.

Figure 3 is a schematic block diagram of the transponder system mounted in each vehicle. The transponder box 99 in the vehicle receives power from car battery through the OBD interface 33a. The transponder box 99 has a GPS receiver such as that produced and manufactured by Garmin International Incorporated. The transceiver can be a radio transceiver produced and manufactured by Freewave Technologies of Boulder, Colorado.

Figure 4 is a schematic diagram of the on-board diagnostic (OBD) circuit for the vehicle-based electronics and transponder. The on-board diagnostic circuit handles such information as speed, acceleration, heading, ignition status, etc. and generates the proper digital signals 96a for delivery to transponder control module 30.

Figure 10 illustrates the general functionality of the vehicle transponder control program software and firmware. The program monitors and logs all in-range vehicles and intersections and manages the data output to the operator display. The core component of the transponder software is the navigation prediction module software task 655. The task uses

position estimates by GPS and other absolute position inputs, and combines data from accelerometers, gyroscopes, tachometers, and heading indicators. This data is then integrated with historical logs. This process, commonly known as dead reckoning, uses accurate (yet possibly intermittent) position reports integrated with time-based inertial navigation data to generate enhanced position estimates. Position information is forwarded to the transponder state and position monitor software task 650. This task monitors vehicle state and diagnostic inputs (such as Code-3) and generates position/state reports to broadcast via the wireless network.

Figure 13 illustrates an example network topology for the communications and operations network. Emergency vehicles 300 and 301 send navigation reports (i.e. GPS) and other data/commands (via wireless connection) to/from intersections and other local vehicles. Preemption-equipped intersections 305, 306, and 307 monitor navigation information from vehicles. Intersections cooperatively and redundantly communicate with each other 320 (via wireless or LAN) to enhance data accuracy and ensure robust communications. Data is also passed along to existing TMC (traffic management center) 330 using existing city LAN communications network 325. If a LAN network is not used, wireless systems can be substituted, such as through FMC 340 (fleet management center) systems. From there, FMC can

forward all data to/from vehicle and TMC.

Figure 14 is a block diagram of the operations software, designed for use in central command centers, mobile command stations, and in individual emergency vehicles. The diagram illustrates the primary functional components of the software. The primary components include algorithmic modules and visual displays for: low-level data activity 405, priority alerts 410, intersections' data 420, vehicles' data 430, and geographic mapping 450. In Figures 15, 16, and 17, both data and displays for these components are shown in an example preemption scenario. This example demonstrates the real-time operations monitoring of a conflict detection scenario, whereby two police vehicles are approaching the same intersection in high priority mode. Figure 15 shows incoming data 461 from vehicles and intersections within the preemption operations communications network 460. Textual status messages are provided on the data status module display 405a. The data status module 405 also maintains a historical record for all low-level communication and data-flow activity. This module 405 relays all verified and priority data messages 406 (i.e. position, preempt, and conflict messages) to the alerts module 410. The alerts module display 410a provides real-time visual notifications of current high-priority events (i.e. active Code-3 vehicles and preempted intersections) and enables rapid analysis of the current

preemption system status.

The alerts module 410 forwards all detailed data 411 to the vehicles and intersections modules 420 and 430. The intersection module display 420a shows real-time detailed intersection data including the traffic light states 421a (phasing) and pedestrian clearance states 421b. Also shown are timing parameters 421c (for example, minimum ETA to intersection for inbound direction) and display data (for example, visual warning signs' states). The vehicle module display 430a shows real-time detailed vehicle data including estimated locations, car types, priority-states, navigation data (such as heading), and other historical information.

All vehicles' and intersections' active data 411 is integrated and overlaid on the mapping module display 450a. The display is an adjustable city map with active units shown as icons, such as vehicle units 431a, 431b and intersection units 432. Visual high-priority alerts, such as conflict detection warnings 433, are logistically overlaid on the map.

A secondary component of the operations software is used for installation and real-time configuration of units 470 as they are added to the preemption network. For intersections, configuration commands 471 include the upload of street grid databases, phase preemption information, and enter/exit distance and timing. For vehicles, configuration commands 471

include ID tags, selection of vehicle type, and sensitivity settings for navigation algorithms. Various test utilities allow the installer to visually monitor the intersection and approaching test vehicles. For instance, the system can be put into the silent preempt mode (no warning signs), or can be manually activated to preempt without a vehicle. The software can communicate directly with a local intersection or vehicle, or can use the local unit's transceiver to talk to the rest of the network.

The operations software can be used to analyze (and optimize) call response times and call response strategies (routes, etc.). It can be used from any location within the range of the network, and can also be integrated into existing call-response centers. The software can also be used for emergency logistics management (i.e. multiple car responses), preventative warnings (i.e. conflict detection), and can also be integrated into existing TMC incident management systems. The system and displays can be accessed via the internet 480 as well. Traffic technicians can use the system to monitor phasing and optimize internal controller programming to match desired preemption settings and behavior. The monitor software is also able to identify potential problems or conflicts in the network using intelligent "sniffer" software utilities. These algorithms watch incoming data to make sure that data is

disseminated in real-time, that data is cohesive and error-free, and that position/state reports are consistent. The system also has the capacity to quickly and autonomously shut off problem vehicle or intersection units. These utilities allow the system to quickly identify anomalies and request maintenance, thereby drastically reducing potentially significant traffic problems.

Thus there has been disclosed improvements to an emergency vehicle traffic signal preemption system. Improvements include providing an autonomous system that is not dependent on intersection being in visual range. The system provides conflict detection and alerts emergency vehicle operators in the area, and provides real-time monitoring of an intersection phase. The real-time monitoring of intersections is indicated by LEDs on a transponder or LCD display in the emergency vehicle that show whether there is a conflict or the intersection being approached is not preempted. The system also includes the improvement of an audio alarm to alert pedestrians who may not be aware of an approaching emergency vehicle for various reasons or are at an angle where visible signs are not clear.

This invention is not to be limited by the embodiment shown in the drawings and described in the description which is given by way of example and not of limitation, but only in

accordance with the scope of the appended claims.

APPENDIX A

The following phrases and definitions are used to describe preemption-related terms, operator-configured parameters, and software-derived calculations. These terms specifically relate to (a) the decision flow diagram in Figure 11, (b) the example preemption scenario shown in Figure 12, and (c) the decision criteria used in the intersection preempt monitor software task:

General Definitions:

"Complete preemption" is the state where a preemption command has been sent to an intersection controller, and the command has been completed such that all PED and traffic lights are "red", except the inbound traffic light for a preempting emergency vehicle which is "green".

"Street segment" is a line (vector) that when combined with other contiguous street segments, represent a street map in the intersection control program software. The segments identify all local streets near or crossing the intersection.

"Critical-inbound" refers to an emergency vehicle that is on a cross street segment, inbound based on its heading, and its ETA or proximity make it eligible for preemption. A vehicle in this state, except in special circumstances, would be preempting the intersection.

"Hysteresis" is a historical dependence statistical

calculation. It uses behavior or rules formed while collecting previous time-based sequenced data to predict future behavior. In the context of this preemption system, hysteresis is used to address such observations as: "if an e-operator successfully preempts a traffic light, the intersection program should be very conservative and cautious before discontinuing the preemption for that vehicle." This basic hysteresis approach is illustrated in Figures 11 and 12. Advanced approaches use tracking and prediction algorithms to more accurately assess vehicle position, e-operator intent, and optimize intersection controller behavior.

Operator-Configurable Values:

"Max-preempt-perimeter" is the maximum distance at which a vehicle is allowed to preempt the local intersection. As example, 3000-ft could be used.

"Street width" is the maximum deviation (distance) allowed between the line-center of a street segment and a vehicle's estimated position. If the calculated difference is less than "street width", the vehicle is considered "on" a street segment. As example, 50-ft could be used.

"Heading error" is the maximum deviation (angle) allowed between the direction of a street segment and a vehicle's estimate heading. If the difference between angles is less than the "heading error", the vehicle is considered to be

moving "along" that street segment. As example, 15-degrees could be used.

"Critical distance" is the distance within which a vehicle is automatically marked as critical-inbound (if heading meets criteria). As example, 200-ft could be used.

"Critical segment" is a boolean value that applies to all street segments; if "yes" then any vehicle "on" that street segment is automatically marked as critical-inbound (if heading meets criteria).

"Max-PED-perimeter" is the distance within which pedestrian-inhibit is enabled to prevent standard PED clearance phases. As example, 2200-ft could be used.

"Min-exit-distance" is the minimum outbound distance past which egress intersection-based warnings are allowed. As example, 30-ft could be used.

"Max-exit-distance" is the maximum outbound distance up to which egress intersection-based warnings are allowed. As example, 100-ft could be used.

"Min-exit-speed" is the minimum speed above which outbound intersection-based warnings are allowed. As example, 5-mph could be used.

"Min-preempt-speed" is the minimum speed above which inbound preemption and inbound intersection-based warnings are allowed. As example, 10-mph could be used.

"Max-latency" is the maximum time between preempt-able messages (see latency-counter description) from the same vehicle before that vehicle is considered inactive. As example, 6-secs could be used.

<u>Software Derived/Calculated Values:</u>

"Max-NAV-error" is the maximum estimated distance error allowed for vehicle-ETA calculations, as determined by dead reckoning algorithms and positioning device specifications. Any error exceeding this factor will invalidate the associated estimated vehicle position. As example, 150-ft could be used.

"Vehicle-ETA" is the minimum estimated ETA (estimated-time-of-arrival) of a vehicle at an intersection, as calculated using the real-time map distance between vehicle and intersection, vehicle speed, vehicle acceleration (based on historical averaging and vehicle type), street type, and expected street conditions (i.e. time-of-day).

"Threshold-lag" is the minimum estimated time that the complete-preemption state must remain steady prior to a preempting vehicle's arrival at an intersection. This calculation is based on the vehicle's speed. The purpose of this factor is to minimize slowing of preempting vehicle. The lag includes threshold-hysteresis (see below).

"Threshold-hysteresis" is a percentage time error included in threshold-lag. When a vehicle preempts an intersection, the

threshold-hysteresis factor resets from 0% to a percentage of the initial vehicle-ETA. For example, 30% could be the default initial setting. Every second thereafter, this percentage is reduced linearly, until 0%. This ensures that once a vehicle is preempting, it is unlikely a temporary vehicle change will disable preemption (i.e. slowing down).

"Time-to-preempt" is the minimum time to achieve complete preemption at an intersection, estimated by the real-time phasing monitor. One of the primary calculations to determine a vehicle's preempt eligibility is if a vehicle's ETA is less than the sum of the time-to-preempt and threshold-lag parameters.

"Latency-counter" is the number of seconds since the last "valid" preempt-able message was received from a given vehicle. Some criteria that would cause the latency counter to increment are: (a) a position report accuracy worse than Max-NAV-error, (b) vehicle not "on" a street segment, (c) low or no vehicle speed, or (d) vehicle heading not inbound.

APPENDIX B

Emergency Vehicle Traffic Signal Preemption System
Vehicle Transponder and Intersection Module Software:
Pseudo-code of Release Versions

```
//
//
//
//
// The system allows emergency vehicles to preempt traffic intersections
// and also provides visual indication (an LED sign) to motorists of
// approaching emergency vehicles.  The system is based on a short-range,
// mobile wireless network with continuous reporting of vehicle state to
// nearby intersections.  The software is written in "C" for the ZWorld
// LP3100 micro-controller.
//
// VEHICLE CONFIGURATION:
//
//  This is the vehicle software component of the EViews Emergency
Vehicle
//  Preemption System.  It determines an emergency vehicle's location and
//  speed, identifies the state of the emergency vehicle (i.e. Code-3),
//  and transmits this information to a network of intersections.  It
also
//  provides feedback to the driver: (1) visual indication of whether the
//  vehicle is currently preempting an intersection, and (2) area mapping
//  data of other nearby emergency vehicles that are in Code-3.
//
// INTERSECTION CONFIGURATION:
//
//  This is the intersection component of the EViews Emergency Vehicle
//  Preemption System. It monitors nearby vehicles in Code-3 and the
//  current timing of all traffic light phases and pedestrian clearance
//  phases.  It uses these parameters to determine when to disable the
//  pedestrian crossing buttons and preempt the traffic signal.  It also
//  broadcasts the current state of the intersection to the nearby Code-3
//  vehicles and other nearby intersections.
//
// MAJOR PARAMETERS USED INCLUDE:

#define   IS_TDMA        1        // is TDMA comm being used?
#define   MASTER_TXMT    1        // is the unit a master (repeater)?

#define   TYPE_VEH       0        // type of vehicle (fire, police, etc)
#define   TYPE_INT       1        // type of intersection (major, minor,
                                  // route, etc)

/******************************************************/
// GENERAL COMM AND DATA
/******************************************************/

#define   MESSAGE_POS    1        // position report
#define   MESSAGE_VID    10       // vehicle ID change
#define   MESSAGE_IID    15       // intersection ID change
#define   MESSAGE_SEG    20       // intersection segment addition
#define   MESSAGE_MSG    30       // textual message
```

```
#define   MESSAGE_DIO    40        // digital I/O info
#define   MESSAGE_IMO    50        // intersection phase info
#define   MESSAGE_ISI    60        // intersection monitor information
#define   MESSAGE_SMP    70        // manual preeemption command
#define   MESSAGE_PSI    80        // vehicle preemption information
#define   MESSAGE_WRT    90        // write parameters to stored data
#define   MESSAGE_CID    100       // change unit ID

#define   OM_POS         1         // output position?
#define   OM_REC         2         // output receipt (of command)?
#define   OM_INT         4         // output intersection info?
#define   OM_EVW         8         // output eviews info?
#define   OM_GSP         16        // use GPS speed?
                                   // (as opposed to vehicle speed)
#define   OM_VOU         32        // use vehicle output icon
#define   OM_CD3         64        // code-3 enabled
#define   OM_TXT         128       // txmt pwr enabled
#define   OM_SGN         256       // use eviews signs?

#define   MAX_VEHICLES_PER_INT    10   // max # of cars per intersection


/////////////////////// VEHICLE CONSTANTS/VARIABLES
//////////////////////

#define   EVEHICLE            1    // is this an emergency vehicle?

#define   VS_CD3              1    // code3
#define   VS_CD2              2    // code2 (silent no sign preempt)
#define   VS_EXT              4    // extension (i.e. bus)

#define   MIN_LEDTIME         205  // div by 50 for secs to hold LED's

#define   MAX_CD3_DELAY       5    // max time Code3 is held (latency)
#define   MAX_CD2_DELAY       5    // max time Code2 is held (latency)
#define   MAX_EXT_DELAY       20   // max time Extension held(latency)

shared    float     Txt_Delay;     // amount of seconds to wait for
                                   // "no data" from OBD before
                                   // shutting off transmitter

#define   MAX_TXT_DELAY       30   // secs to wait for OBD to come
                                   // online before turning off OBD

shared    float     StopTime;

#define   MAX_STOPTIME        25   // max times to allow vehicle stop
                                   // and still active
#define   MAX_STOPEVEH        10
#define   MAX_STOPEXT         20


//////////////////// INTERSECTION CONSTANTS/VARIABLES //////////////////

#define   PRE_EMERGENCYVEHICLE    10   // sign options (VMS)
#define   PRE_POLICEPURSUIT       6
#define   PRE_CLEARINTERSECTION   5
#define   PRE_NOLEFTTURN      1
```

```
#define   PRE_NORIGHTTURN     2

#define   MIN_DWLK_SOLID      1.5  // read PED min time

float     DWk_Solid[8];            // time that dont-walk been solid
float     Ped_Clear[8];            // time that dont-walk has been
blinking
float     Yel_Timer[8];            // time at which Yellow last toggled ON
int       T_YToR[8];               // amount time for yellow light on
phase
int       T_WToR[8];               // amount time for maximum ped on phase

int       MinTimeToInt;            // closest vehicle's ETA to
intersection
int       MinDistToInt;            // closest vehicle's distance to
                                   // intersection

#define   MAX_PREEMPT_WINDOW 6     // hystersis window for preemption (so
                                   // borderline triggering is avoided)

int       PreSigStat[4];           // current preempt status (includes
type
                                   // of preempt)
int       LastEviewUpdate;

#define MAX_EVIEWUPDATE       10
#define MAX_PEDINHIBIT        10   // min hold time once ped preempt
starts
#define MAN_PEDPREEMPT        100  // ID for source on manual ped preempt

#define MAX_EXTACTIVATION     6    // min hold time once extension starts
                                   // (i.e. bus)

#define INT_PERIMETER         3000 // intersection will not preempt for
any
                                   // vehicle outside this perimeter
#define PED_PERIMETER         2200 // distance at which ped inputs
                                   // are prevented
#define EXT_PERIMETER         500  // distance at which vehicle-extension
                                   // is actuated

#define   IS_SIGREAD          0    // is signal reading active?
#define   CFG_TIMETOPREEMPT   20.0 // if signal reading not active, what
is
                                   // min ETA time to use for preemption
                                   // (after critical distance)
#define   IS_SIGLEGAL         0    // are EViews signs activated
                                   // based on signal condition
                                   (legality)?

#define   LOW_PREEMPT         5    // lower end of bracket for low
priority
                                   // (extension) vehicles

#define   CRITICAL_DISTANCE   200  // distance under which ETA is ignored
                                   // and vehicle automatically preempts
                                   // (commit distance)
#define   MAX_TIMETOPREEMPT   30
```

```
#define   MAX_LATENCY        30

typedef struct SegmentType_Tag {  // position information
     float          Lat1;
     float          Lon1;
     float          Lat2;
     float          Lon2;
     float          Dist;
     float          Head;
     int       Loc;
     int       IsCritical;
} SegmentType;

#define   MAX_SEGMENTS       30   // maximum number of street segments
                                  // accepted per intersections

typedef struct SD_Tag {

     long      UnitID;            // unique unit ID
     int       VehType;           // type of vehicle
     int       StreetWidth;       // allowable error in street width (ft)
     int       Latency;           // allowable delay between updates
                                  // before vehicle is marked inactive
                                  // (secs)
     int       HeadingSpan;       // allowable error in heading
     int       MaxPosLatency;     // max time to use dead reckoning w/o
                                  // a valid Pos (i.e. GPS) lock
     int       DeltaNorth;        // used to calibrate intersection to
                                  // north
     int       PreemptMode;       // determines how to handle preempts
     float     TimeToPreempt;     // maximum seconds to preempt all
phases
     int       ExitDistance;      // determines time to output outgoing
                                  // icons
     int       ThresholdLag;      // minimum time to preempt before
                                  // intersection threshold
     int       SourceToPRE[4];    // orientation of preemption phases
     int       OutputMode;        // output settings
     int       NumSegments;       // number of street segments in memory
     float     InteLat;           // longitude for intersection center
     float     InteLon;           // latitude for intersection center
     float     DeltaLat;          // calibration delta for 1 foot at int
     float     DeltaLon;
     SegmentType
               Segments[];        // street segments

} SD_Type;

// preempt mode
#define   ALL_RED            0    // 0=ALL SIGNALS GO RED
#define   ONE_GREEN          1    // 1=SIGNAL IN VEHICLE(S) DIRECTION
GOES
                                  // GREEN MULTIPLE VEHICLES, MULTIPLE
                                  // DIRECTIONS ALL RED)


/*******************************************************/
// MAIN
```

```
/*****************************************************/

main()
{
     InitBoard();
     InitComm();
     InitConfig();

     // hit watchdog
     hitwd();

     // assign type of hardware
     #if IS_VEHICLE
          Vehicle_Init();
     #else
          Intersection_Init();
     #endif

     // run background task always
     backgnd();
}

////////////////////////////////////////////////////////////////////////////
// Config_Init
// Determine if valid parameters are in EPROM; if not, load defaults
// Only called if system is reprogrammed or power is lost
////////////////////////////////////////////////////////////////////////////

Config_Init()
{
     StoredExists=False;
     if (EPROM_Exists) {
          LoadEPROMData(StoredExists);
     }
     VerifyStoredData(!StoredExists);
}

////////////////////////////////////////////////////////////////////////////
/
// background task runs when no other task is running
////////////////////////////////////////////////////////////////////////////
/

backgnd()
{
     while (True) {
          // do nothing except hit watchdog timer
          hitwd();
     }
}

//********************************************************************//
//                          INTERSECTION ROUTINES                      //
//********************************************************************//

////////////////////////////////////////////////////////////////////////////
// Task_IntMonitor
// Monitors all incoming traffic signals to determine preemption timing
```

```
// Also sends out periodic "preempt" status signals to all cars
/////////////////////////////////////////////////////////////////////////////

Task_IntMonitor()
{
#if !IS_VEHICLE
     for each Phase {
          // read current state of traffic signal and ped signal
          ReadPhaseInfo(CurRed,CurYel,CurGrn,CurWlk);

          // dynamically determine ped timing & ped clearance for phase
          DeterminePEDTiming(CurWlk);

          // calculate expected clearance time for this phase
          DetermineSignalTiming(CurRed,CurYel,CurGrn);
     }

     CurrentClearanceTime = Max(clearance time of all phases);

     if (OutputEnabled)
          // if output enabled, send information to all units every
second
          SendInfotoNetwork(phasing information);

     if (Preempting)
          // adjust hysterisis window (window expanded when vehicle
starts
          // preemption, and slowly collapsed) prevents threshold
          // triggering ON/OFF if vehicle is on the border of preemption
          DecreaseSizeOfPreemptWindow;

     // if preempted, send current PreemptVehicles to all units at 1-Hz
     PMessage_SendPreemptVehicles();
#endif
}

/////////////////////////////////////////////////////////////////////////////
/
// Send a sentence to all signs
/////////////////////////////////////////////////////////////////////////////
/

Eview_SendSentence()
{
#if !IS_VEHICLE
     hitwd();
     CreateVMSMessage(Eviews(data));
     SendVMSMessage(SignID);
#endif
}

/////////////////////////////////////////////////////////////////////////////
// Intersection_Preempt
// Changes current state of preemption for PreemptMonitor
// Handles state of all preempting vehicles
/////////////////////////////////////////////////////////////////////////////

Intersection_Preempt()
```

```
{
#if !IS_VEHICLE
     // if car is active, determine if car is already registered;
     // otherwise, create new entry for new car
     CurrentCar = FindVehicleInfo(VehicleID);

     if (CarInactive)
          // if car inactive (code off)
          DeleteVehicleFromList(VehicleID);
     else
          // store vehicle data (ID, direction, state, speed, etc)
          StoreVehicleData(CurrentCar);
#endif
}

////////////////////////////////////////////////////////////////////////////
// Controls DIO for signal preemption (including low priority modulation)
////////////////////////////////////////////////////////////////////////////

Task_SigPreControl()
{
#if !IS_VEHICLE
     // dynamically reads traffic signal state at 10Hz from hardware
input
     ReadTrafficSignals(SignalMatrix);
#endif
}

////////////////////////////////////////////////////////////////////////////
// Starts/monitors traffic signal preemption and then
// starts/maintains eviews sign preemption
// (based on intersection conditions)
////////////////////////////////////////////////////////////////////////////

Task_PreemptMonitor()
{
#if !IS_VEHICLE
     // init Eviews settings
     InitEViewsMem(OFF);

     for (all vehicles)
          // review current preemption vehicle list & activate/deactivate
          // VMS icons
          SetEViewsMem(CurrentVehicle, VehicleDirection,
                VehicleActiveStatus);

     for (all main phases) {
          if (VehicleActive(CurrentPhase))
               // set all traffic preempt lines using vehicle list
               SetControllerPreempt(CurrentPhase);
     }

     if (PEDTriggered or IntersectionIsPreempted)
          // if PED timer is active or intersection is actively
          // preempting, prevent PED input
          DisablePED();

     if (LastEviewUpdate<=MAX_EVIEWUPDATE) {
```

```
            // if signal was preempted in last update seconds,
            // determine signal state for eviews sign
            #if (SignalReadActive)
                 // if signals are available and signal rules are in effect
                 // for warning sign, determine legality
                 SetLegalCondition(IllegalCondition, PhaseInfo);
            #endif

            if (not IllegalCondition)
                 // IF NOT ILLEGAL SIGNAL CONDITION, transmit information
                 // to local signs
                 Eview_SendSentence();

            if (EViewsOutputEnabled)
                 // if enabled, send eview sign information to other
                 // units on network
                 SendInfotoNetwork(sign information);
     }
#endif
}

/////////////////////////////////////////////////////////////////////////////
// Intersection_Init
// Initialize intersection variables
/////////////////////////////////////////////////////////////////////////////

Intersection_Init()
{
#if !IS_VEHICLE
     // initialize all phases, preempt lines, transmit lines, etc
     IntInitParameters();

     // initialize vehicle preempt list
     VehInitParameters();

     // schedule traffic light monitor task to run every 1/2 sec
     Task_IntMonitor();

     // start preempt monitor
     Task_PreemptMonitor();
#endif
}

/////////////////////////////////////////////////////////////////////////////
// Intersections_Update
// Determines if a vehicle is within the "preempt" boundaries of
// the intersection
/////////////////////////////////////////////////////////////////////////////

Intersection_Update()
{
#if !IS_VEHICLE
     // compute distance as crow flies to figure intersection point
     CrowDistCarToInt = ComputeLatLonDist(PositionInfo);

     if (Vehicle is (Code3 or Code2 or Extension) and
           CrowDistCarToInt<INT_PERIMETER) {
          // if car in code3 or code2, and car within perimeter distance,
```

```
            // determine proximity
            for (all road segments)
                DetermineCarProximityToIntersection(Distance);

            if (Distance within Preempt boundaries)
                // vehicle is within preempt rules, send closest
                // segment information
                Intersection_Preempt(Enable for Current Road Segment);
        }
        else {
            // code-3 disabled, eliminate code3
            Intersection_Preempt(Disable for CurrentVehicle);
        }
    #endif
    }

    //*********************************************************************//
    //                          LOCATION ROUTINES                          //
    //*********************************************************************//

    ////////////////////////////////////////////////////////////////////////
    // Initialize ports, sets up position buffers, and starts
    // positioning tasks
    ////////////////////////////////////////////////////////////////////////

    Vehicle_Init()
    {
    #if IS_VEHICLE
        // initialize Vehicle Indicators
        InitVehicleVisualDisplay();

        // open serial channel
        InitVehiclePorts();

        // start position reading
        Task_CalculateRealTimePosition();

        // schedule dead reckoning (supplemental)
        Task_DeadReckoning();

        // schedule dead reckoning
        Task_VehicleVisualDisplay();
    #endif
    }

    ////////////////////////////////////////////////////////////////////////
    // If positioning is current and valid (i.e. GPS > 3 sats), output
    // current info otherwise, if time within MaxLatency, compute dead
    // reckoning using speed and heading of vehicle
    ////////////////////////////////////////////////////////////////////////

    Position_SendAccuratePosition()
    {
    #if IS_VEHICLE
        GetCurrentPosition(PositionInfo);
        if (PositionInfo is Old) {
            // if lag less or equal to MaxLatency, use dead reckoning pos
            GetDeadReckon(PositionInfo);
```

```
        }

        if (CurrentVehicle is not stopped longer than threshold)
            VehState=ActiveCode;
        }

        SendInfotoNetwork(Vehicle Position & State Information);
    #endif
    }

    ////////////////////////////////////////////////////////////////////////////
    // Indefinitely reads position data (i.e. from GPS serial port)
    ////////////////////////////////////////////////////////////////////////////

    Task_CalculateRealTimePosition()
    {
    #if IS_VEHICLE
        // indefinitely calculate vehicle position
        while (True) {
            CalculateBestPosition(Default=GPS);
        }
    #endif
    }

    ////////////////////////////////////////////////////////////////////////////
    // Computes current dead reckoning position
    ////////////////////////////////////////////////////////////////////////////

    Task_DeadReckoning()
    {
    #if IS_VEHICLE
        // read current speed (kph)
        ReadSpeed(OBDInfo);

        // if OBD disabled, assume car is off
        if (OBDInfo.Disabled)
            // if OBD disabled, shut off transmitter
            TxmtTurn(OFF);

        // compute distance travelled since last update (ft/sec)
        DistanceTraveled = IntegrateSpeed(SpeedHistory);

        // get current heading
        Heading_Read();

        // read code staus, handle timing to indicate when last code was
    seen
        ReadCodeStatus(VehType, CodeMatrix);

        if (OBDSpeed>0 Or PositionSpeed>0 Or CodeChange)
            // if vehicle is moving or code3/code2/ext was just turned on,
            // force fresh code call
            MakeCurrentCode(CodeMatrix);
        else
            // if vehicle is stopped, increment stop counter
            DelayCurrentCode(CodeMatrix);
    #endif
    }
```

```
//*********************************************************************//
//                          COMMUNICATION ROUTINES                      //
//*********************************************************************//

/////////////////////////////////////////////////////////////////////////
// Comm_DataMoveValue
// Adds a new data value to data message
/////////////////////////////////////////////////////////////////////////

Comm_DataMoveValue()
{
     SelectDataType(DataSize);
     AssignDataValue(DataSize, DataValue, OperationType);
}

/////////////////////////////////////////////////////////////////////////
////////
// Sends/Receives POS message type
/////////////////////////////////////////////////////////////////////////
////////

PMessage_POS()
{
     Comm_DataMoveValue(VehType,VehState,GSpeed,VSpeed,Lat,Lon,
                              PosQuality,GHeading,VHeading);

     if (DataMode==WRITE)
          // if vehicle, send position info to network
          SendInfotoNetwork(VehicleInfo);

     if (DataMode==READ)
          // if intersection, update preemption status for
          // notifying vehicle
          Intersection_Update();
}

/////////////////////////////////////////////////////////////////////////
////////
// Sends/receives intersection line segment message type
/////////////////////////////////////////////////////////////////////////
////////

PMessage_SEG()
{
#if !IS_VEHICLE
     Comm_DataMoveValue(C1Lat,C1Lon,C2Lat,C2Lon,
                         Distance,Heading,Location,IsCritical);

     if (DataMode==READ)
          // if Intersection, read in all street segments and config info
          // for permanent store
          StoreMapAndConfig();
#endif
}

/////////////////////////////////////////////////////////////////////////
// Executes a manual preempt command
```

```
////////////////////////////////////////////////////////////////////////////

PMessage_SMP()
{
#if !IS_VEHICLE
     Comm_DataMoveValue(Source,Direction,VehType,VehState);

     if (DataMode==READ)
          // enable manual (remote) preempt of phase/ped signals
          Intersection_Preempt();
#endif
}

////////////////////////////////////////////////////////////////////////////
////////
// Maintains preemption LED statusm in vehicle
////////////////////////////////////////////////////////////////////////////
////////

Task_VehicleVisualDisplay()
{
#if IS_VEHICLE
     while (True) {
          // indefinitely convert vehicle status and collision avoidance
          // information into visual in-car indicators (LED's, PDA's, or
          // PC) - maps, text warnings, LED's
          OutputLEDInfo(LEDMatrix);
          OutputPCInfo(PCInfo);
          OutputPDAInfo(PDAInfo);
     }
#endif
}

////////////////////////////////////////////////////////////////////////////
////////
// Handles currently active vehicle preempts
////////////////////////////////////////////////////////////////////////////
////////

PMessage_SendPreemptVehicles()
{
     Comm_DataMoveValue(All Vehicles Listed);

     if (DataMode==READ) {
          #if IS_VEHICLE
               // generate visual display based on all actively
preempting
               // vehicles if only one vehicle is preempting and it is
               // this vehicle, light green LED if more than one vehicle
               // is preempting and it includes this vehicle,
               // light yellow LED
               VehicleVisualDisplayUpdate(AllActiveVehicleStatus);
          #endif
     }
     else {
          #if !IS_VEHICLE
               // notify all units of those cars who have preempted
               // in last 2 seconds
```

```
                SendInfotoNetwork(AllActiveVehiclesInfo);
            #endif
        }
    }

    ///////////////////////////////////////////////////////////////////////////
    // Outputs intersection calculated information
    // Includes derived parameters (last trigger per phase, etc)
    ///////////////////////////////////////////////////////////////////////////

    PMessage_ISI()
    {
    #if !IS_VEHICLE
        Comm_DataMoveValue(ISI_Type,IntParam1,IntParam2,
                                    IntParam3,IntParam4,IntParam5,
                                    IntParam6,IntParam7,IntParam8);

        if (DataMode==WRITE)
            SendInfotoNetwork(IntersectionPhaseInfo);
    #endif
    }

    ///////////////////////////////////////////////////////////////////////////
    // Outputs intersection monitor information
    // Includes red,grn,yel phasing and red, yellow, ped clearance
    ///////////////////////////////////////////////////////////////////////////

    PMessage_IMO()
    {
    #if !IS_VEHICLE
        Comm_DataMoveValue(I_Phase,I_SignalType);

        if (DataMode==READ)
            SendInfotoNetwork(IntersectionMonitorInfo);
    #endif
    }

    ///////////////////////////////////////////////////////////////////////////
    // Sends/receives IID message type
    // Intersection configuration information
    ///////////////////////////////////////////////////////////////////////////

    PMessage_IID()
    {
    #if !IS_VEHICLE
        Comm_DataMoveValue(StreetWidth,Latency,HeadingSpan,OutputMode,
                           TimeToPreempt,DeltaNorth,PreemptMode,
                           ExitDistance,ThresholdLag,PreemptOrient);

        if (DataMode==WRITE)
            // send information back to requestor
            SendInfotoNetwork(IntersectionConfigInfo);

        if (DataMode==READ)
            StoreIntersectionConfigInfo(IntersectionConfigInfo);
    #endif
    }
```

```
////////////////////////////////////////////////////////////////////////////
// Sends/receives VID message type
////////////////////////////////////////////////////////////////////////////

PMessage_VID()
{
#if IS_VEHICLE
     Comm_DataMoveValue(VehType,OutputMode,MaxPosLatency);

     if (DataMode==WRITE)
          // send information back to requestor
          SendInfotoNetwork(VehicleConfigInfo);

     if (DataMode==READ)
          // set vehicle config info
          SetVehicleConfigInfo(VehicleConfigInfo);
#endif
}

////////////////////////////////////////////////////////////////////////////
// Allows change of Unit ID
////////////////////////////////////////////////////////////////////////////

PMessage_CID()
{
     Comm_DataMoveValue(NewID,UnitType);

     if (DataMode==READ)
          SetVehicleIDInfo(VehicleIDInfo);
}

////////////////////////////////////////////////////////////////////////////
// Write stored information to EPROM
////////////////////////////////////////////////////////////////////////////

PMessage_WRT()
{
     WriteStoredData();
}

////////////////////////////////////////////////////////////////////////////
// Sends/receives string message type
////////////////////////////////////////////////////////////////////////////

PMessage_MSG()
{
     Comm_DataMoveValue(MessageLen,Message);

     if (DataMode==WRITE) {
          SendInfotoNetwork(MessageInfo);
}

////////////////////////////////////////////////////////////////////////////
// Sends/receives string message type
////////////////////////////////////////////////////////////////////////////

PMessage_DIO()
{
```

```
        Comm_DataMoveValue(Channel,Operation,Value);

        ReadDirectPortDigitalIO(PortDIOInfo);

        SendInfotoNetwork(PortDIOInfo);
    }

    /////////////////////////////////////////////////////////////////////////
    // Parses data from a packet and calls appropriate function to handle
    // the data
    /////////////////////////////////////////////////////////////////////////

    Comm_ParseData()
    {
        SelectMessage(MessageType);
    }

    /////////////////////////////////////////////////////////////////////////
    // Packs data and sends to comm
    /////////////////////////////////////////////////////////////////////////

    SendInfotoNetwork(Data);
    {
        Packet=BuildPacket(Marker, Length, Checksum, MessageType, PacketID,
                            SourceID, DestinationID, Data);

        if (CommIsTDMA)
            AddTDMAHeader(Packet);

        // send packet to transceiver (wireless net)
        SendPacketToTransceiver(Packet);

        // send packet out local port
        SendPacketToLocalSerial(Packet);
    }

    /////////////////////////////////////////////////////////////////////////
    // Receives packet info, unstuffs information, parses packet info,
    // and then requests processing of data message
    /////////////////////////////////////////////////////////////////////////

    Task_ReceivePacket()
    {
        while (True) {
            Data=ReadPacket(Marker, Length, Checksum, MessageType,
                            PacketID, SourceID, DestinationID, Packet);
            Comm_ParseData(Data);
        }
    }

    /////////////////////////////////////////////////////////////////////////
    // Indefinitely reads all incoming messages from the transceiver
    /////////////////////////////////////////////////////////////////////////

    indirect
    Task_CommRead()
    {
        while (True) {
```

```
                Data = ReadLowLevelComm(IncomingPorts);
                if (UnitIsMasterNode)
                     // if unit is considered a master node in the network,
                     // repeat the message to all local units
                     SendInfotoNetwork(Data, REPEAT);
           }
      }
      //  End of Code
```